A SUGGESTED COMPUTATION FOR MAXIMAL MULTI-COMMODITY NETWORK FLOWS

L. R. Ford, Jr.
D. R. Fulkerson

P-1114

Revised
March 27, 1958

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

SUMMARY

A simplex computation for an arc-chain formulation of the maximal multi-commodity network flow problem is proposed. Since the number of variables in this formulation is too large to be dealt with explicitly, the computation treats non-basic variables implicitly by replacing the usual method of determining a vector to enter the basis with several applications of a combinatorial algorithm for finding a shortest chain joining a pair of points in a network.

A SUGGESTED COMPUTATION FOR MAXIMAL MULTI-COMMODITY NETWORK FLOWS

Introduction

A problem of some importance in applications of linear programming is the determination of maximal multi-commodity flows in networks. For example, some of the linear programming problems which have been proposed recently by Juncosa and Kalaba in their studies of communication networks [5] can be cast in this form. Straightforward application of the simplex method to such problems is usually not feasible, since even small networks may generate linear programs which are too large for present machine capacity. What is needed are specialized computing schemes that take advantage of the structure of such problems. For the single commodity case, various easy computations are known [1,3,4], but the multi-commodity problem has remained relatively unexplored.

Consideration of simple examples makes it appear that the multi-commodity flow problem is considerably more complex than the single commodity one. Certainly the nice combinatorial features of the single commodity case are lost in the generalization - simplex bases (for any formulation of the problem known to us) are not triangular, hence addition and subtraction do not suffice to solve such problems by the simplex method, the max flow min cut theorem, true for single commodity networks, is false [3], and no simple-minded modification of the labeling process [4] seems to work.

The purpose of this note is to suggest a computation which makes some use of the structure of one formulation of the multi-commodity problem within the framework of a simplex computation. For this particular formulation, the matrix of the linear program is the incidence matrix of arcs vs. all chains joining sources and sinks for the various commodities, and thus the number of variables is too large to be dealt with explicitly. The suggested computation treats non-basic variables implicitly by replacing the "pricing" operation of the simplex method (i.e. the determination of a vector to enter the basis) with several applications of a combinatorial algorithm for finding a shortest chain joining a pair of points in a network.

1. Arc-chain Formulation

Let $A_1, \ldots, A_m$ be a list of the arcs of the network, $C_1, \ldots, C_n$ a list of all chains that join, for the various commodities, all the sources for a commodity with all sinks for the same commodity, and let $A = (a_{rs})$ be the $m \times n$ incidence matrix of arcs vs. commodity chains:

$$(1) \qquad a_{rs} = \begin{cases} 1 & \text{if } C_s \text{ contains } A_r, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, for example, if the network is that of Fig. 1, with sources P_1, P_2, sink P_3 for one commodity, and source P_4, sink P_1 for a second commodity, the matrix A is as shown in Fig. 2.

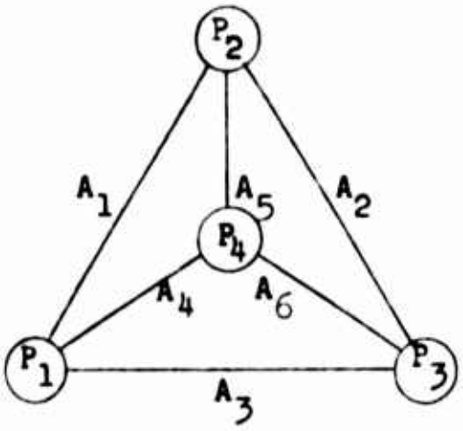

Fig. 1

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
A_1			1		1			1		1		1			1
A_2			1	1		1								1	1
A_3	1							1	1				1	1	
A_4		1		1					1	1	1				
A_5				1	1		1		1			1		1	
A_6		1			1		1			1			1		1

Commodity 1 Commodity 2

Fig. 2

If we let x_s, $s = 1, \ldots, n$, denote the amount of commodity flow along C_s, and b_r the flow capacity of A_r, then the multi-commodity maximal flow problem is represented by the linear program:

(2) $$\text{maximize} \quad \sum_{s=1}^{n} x_s$$

subject to the constraints

(3) $$\sum_{s=1}^{n} a_{rs} x_s + x_{n+r} = b_r,$$

$$x_1, \ldots, x_{n+r} \geq 0.$$

The assumption in (2) that commodities are valued equally is not essential to the method we propose, as will be clear from our discussion in the following section. Another thing we wish to point out is that it is immaterial whether the problem involves directed or undirected arcs. Thus, for example, if there are "one-way streets," or if, in a communication network, say, it is desired to place an upper bound on the number of messages that can be transmitted from P_i and P_j, and an upper bound on the messages that can be sent from P_j to P_i, one considers two arcs, one from P_i to P_j, the other from P_j to P_i, and directed chains from sources to sinks.

Since the number of chains is usually very large in practical applications, the arc-chain formulation of the problem might seem to be impossible to deal with computationally. Indeed, the enumeration of all chains from commodity sources to sinks in a network of moderate size would be a lengthy task, to say the least. Fortunately, there is no need to write down the entire matrix A, since the selection of a variable entering the basic set at any stage of the simplex computation (or the

recognition that a basis is optimal) can be accomplished without explicit knowledge of the non-basic column vectors of A. All we need is the basis $B = (b_{rj})$ (or its inverse), a square submatrix whose order is the number m of arcs in the network, to compute the simplex multipliers $\alpha_r (r = 1,\ldots,m)$ satisfying, for $j = s_1, \ldots, s_m$,

$$(4) \qquad \sum_{r=1}^{m} \alpha_r b_{rj} = \begin{cases} 1 & \text{if } j \leq n \\ 0 & \text{if } j > n. \end{cases}$$

We can then find a vector to bring into the basis (or prove that the current basis is optimal) by the method of the next section. Once such a vector has been found, determination of the vector leaving the basis is accomplished in the usual way.

2. A Shortest Chain Algorithm

Suppose we have computed the α_r in (4) corresponding to a particular basis B. If some α_r is negative, then the variable x_{n+r} may be introduced into the basic set with possibly an increase in the form (2), that is, the unit vector having 1 in the r-th position, zeros elsewhere, can be brought into the basis. (It may be that this vector also represents a one-arc chain for some commodity; in this case, a bigger increase in (2) might result, of course, by taking the latter interpretation.)

Assume, therefore, that a stage has been reached in the computation where all α_r are non-negative. In this case, the algorithm described below, which makes no use of the full

incidence matrix A, can be used either to locate a column vector of A (i.e. a commodity chain in the network), that may be brought into the basis, or to prove that the current basis is optimal.

Let us interpret the α_r as lengths of the arcs. We wish to find a chain C_s, if one exists, whose length

$$\sum_{r=1}^{m} \alpha_r a_{rs}$$

is less than one, the coefficient of x_s in (2). Thus, it suffices to locate, for each commodity, a shortest chain from the commodity sources to its sinks. If each of the chains thus selected has length at least one, the basis is optimal. Otherwise, a column vector of A corresponding to one of these chains may be introduced into the basis.

The problem of locating a shortest chain from one set of nodes to another set of nodes in a network can be reduced to a standard transshipment problem [6], and may consequently be solved in various simple ways; see [2] and [6], for example. The algorithm we describe is that of [2]. (In [2], the problem is considered to be that of finding a shortest chain from one node to another; to reduce our problem to this one, simply join each node of the first set to a new node by an arc of length zero, and similarly for the other set. We shall give a description which does not involve this device explicitly, however.)

Let the set of sources for one commodity be S, the sinks for the commodity T, and suppose the nodes of the network are

$P_1, \ldots, P_N$. Let ℓ_{ij} denote the length of the arc joining P_i and P_j, i.e. if the arc A_r joining P_i and P_j corresponds to the simplex multiplier α_r, set $\ell_{ij} = \alpha_r$. (If arcs are directed, then we let ℓ_{ij} denote the multiplier corresponding to the arc from P_i to P_j, hence in this case we may have $\ell_{ij} \neq \ell_{ji}$, whereas in the undirected case, $\ell_{ij} = \ell_{ji}$.) Initially assign to each node P_i a number π_i as follows:

$$\pi_i = \begin{cases} 0 & \text{for } P_i \in S \\ \infty & \text{otherwise.} \end{cases}$$

Now scan the network for an arc P_iP_j such that

$$\pi_i + \ell_{ij} < \pi_j.$$

Replace π_j by $\pi_i + \ell_{ij}$ if such an arc is found. Continue this process. Eventually no such arcs can be found; then the number π_i represents the length of a shortest chain from S to P_i, for all i. In particular, the smallest π_i, for $P_i \in T$, is the length of a shortest chain from S to T. Let π_k be the smallest such. To find a chain from S to T of length π_k, look for an arc P_jP_k such that $\pi_j + \ell_{jk} = \pi_k$, then search for an arc P_iP_j such that $\pi_i + \ell_{ij} = \pi_j$, and so on. Eventually a node of S is reached, and the desired chain has been traced out (in reverse).

If in the process of locating shortest chains from commodity sources to sinks, for the various commodities, one is found of length less than one, we recommend that the corresponding column

vector of A be introduced into the basis immediately, rather than repeating the shortest chain algorithm a number of times in order to use the usual criterion for selection of a vector to enter the basis.

We point out that the reason for getting rid of negative multipliers α_r before using the shortest chain algorithm is that the algorithm may not work if arcs have negative lengths.

To start the simplex computation, one can of course begin with the basic variables $x_{n+1}, \ldots, x_{n+r}$, corresponding to the zero flow.

3. Concluding Remarks

Except for hand computation of a few small problems, we have no computational experience with the proposed method. Whether the method is practicable for a problem involving, say, 50 nodes, 100 arcs, and 20 commodity source-sink sets $S_1,T_1,\ldots, S_{20},T_{20}$, is a question which can be settled only by experimentation. It would certainly be more practicable in this case than straightforward application of the simplex method to a node-arc formulation of the problem, since in the latter formulation there would be roughly 1100 equations in 2100 variables, and hence the basis matrices would be much too large, whereas in the suggested method, the basis matrices would be 100 x 100, and at most 20 applications of the shortest chain algorithm would be necessary on each simplex iteration. How many simplex iterations might be required is another matter, though. The incidence matrix A for such a problem could have

many thousands of columns. On the other hand, there would probably be many column vectors of A dominated by others, in the sense that, for a given commodity (or for different commodities in the equal value case), if one chain C is a subset of another chain C', then C' can be ignored. (For instance, the chain C_1 of Fig. 2 dominates C_8 and C_9.) The shortest chain method takes care of such dominances automatically.

A more serious consideration is how to handle the case of limited supplies of commodities in such a problem. For example, suppose that in the two commodity maximal flow problem corresponding to the matrix of Fig. 2, there is an amount a_1 of commodity 1 at P_1, an amount a_2 of commodity 1 at P_2, and an amount a_4 of commodity 2 at P_4. We can reduce this to a problem of the same type as before by introducing three new directed arcs and nodes as follows: A_1' from P_1' to P_1 with capacity a_1, A_2' from P_2' to P_2 with capacity a_2, and A_4' from P_4' to P_4 with capacity a_4. We then take P_1', P_2' as sources for commodity 1, and P_4' as the source for commodity 2. However, in the hypothesized large network with 20 commodities, the number of such new arcs would be $\sum_{i=1}^{20} n_i$, where n_i is the number of nodes in S_i, and since each new arc increases the size of basis matrices by one, this might take the problem out of range of present computing machines.

REFERENCES

1. Dantzig, G. B., and D. R. Fulkerson, "Computation of Maximal Flows in Networks," Naval Res. Logs. Quarterly, Vol. 2, No. 4, Dec. 1955.

2. Ford, L. R. Jr., Network Flow Theory, The RAND Corporation Paper P-923, August 14, 1956.

3. Ford, L. R. Jr., and D. R. Fulkerson, "Maximal Flow Through a Network," Can. J. Math., Vol. 8, pp. 399-404, 1956.

4. Ford, L. R. Jr., and D. R. Fulkerson, "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem," Can. J. Math., Vol. 9, pp. 210-218, 1957.

5. Kalaba, R. E., and M. L. Juncosa, "Optimal Design and Utilization of Communication Networks," Management Science, Vol. 3, No. 1, October 1956.

6. Orden, A., "The Transshipment Problem," Management Science, Vol. 2, No. 3, April 1956.